Image & Video Processing

Assignment 4 - Color and Multi-scale Representations

Albert Cerfeda

Contents

Università della Svizzera italiana
Faculty of Informatics

29 May 2023
Università della Svizzera italiana
Faculty of Informatics
Switzerland

1 Color Palette Extraction from Images [7 points]

1.1 Linear RGB and sRGB Color Palettes

```
function [img_clustered, img_palette, img_layers] = clusterColors(img, n_clusters)
    % 1. Cluster
    A = reshape(img, [], 3);
    [cluster_idx, cluster_center] = kmeans(A, n_clusters);

    % Compute clustered image
    img_clustered = reshape(cluster_center(cluster_idx, :), size(img, 1), size(img, 2), 3);
    % Matrix of cluster indices for corresponding image pixel
    cluster_idx = reshape(cluster_idx, size(img, 1), size(img, 2), 1);

    % 2. Separate image into palette layers
    color_square = zeros(size(img));
    img_palette = {};
    img_layers = {};
    for i = 1:n_clusters
        % Create a squared image with a uniform color using function repmat
        color = cluster_center(i, :);
        color_square(:,:,1) = repmat(color(1), size(img, 1), size(img, 2));
        color_square(:,:,2) = repmat(color(2), size(img, 1), size(img, 2));
        color_square(:,:,3) = repmat(color(3), size(img, 1), size(img, 2));
        layer = color_square .* (cluster_idx == i);

        img_palette{i} = color_square;
        img_layers{i} = layer;
    end
end
```

Figure 1: Matlab code performing color quantization

1.2 CIELab Color Palette

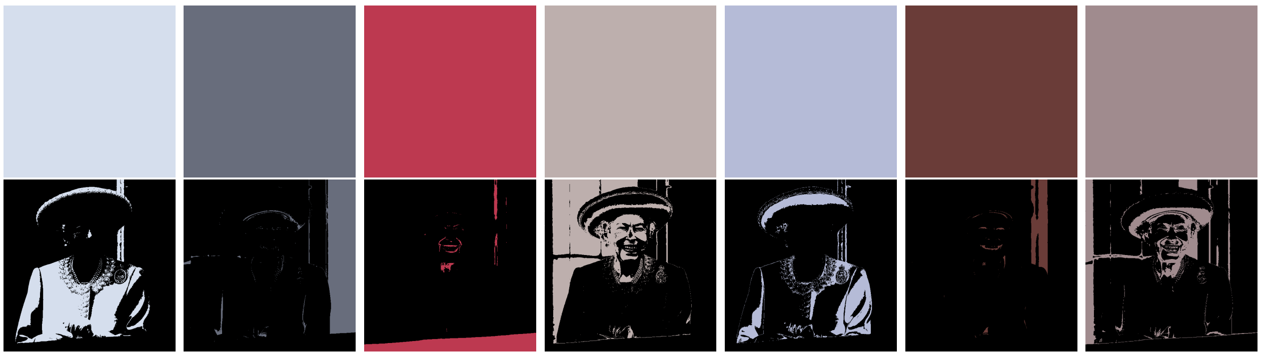
2 Color Quantization and lookup tables (LUTs) [3 points]

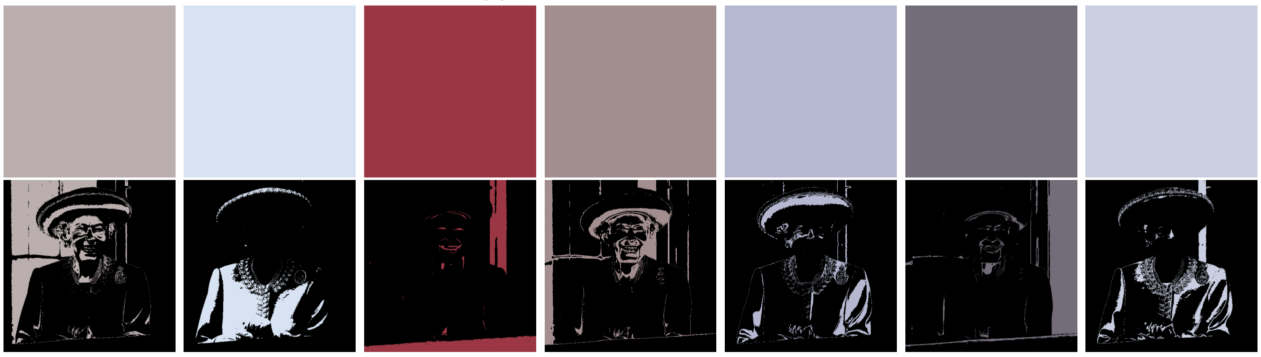
3 Gaussian and Laplacian Pyramids [4 points]

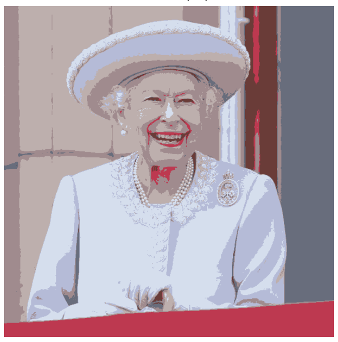
4 Hybrid Images [3 points]

5 Theory: Hybrid Images [3 points]

6 Bonus: Create your own Instagram Filter! [10 points]


(a) RGB Palette and Layers


(b) sRGB Palette and Layers


(c) RGB Clustered image

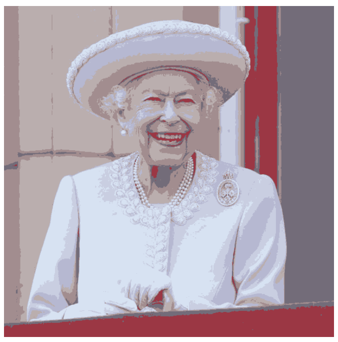
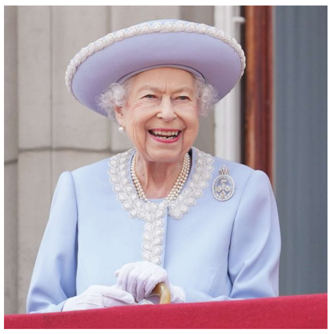
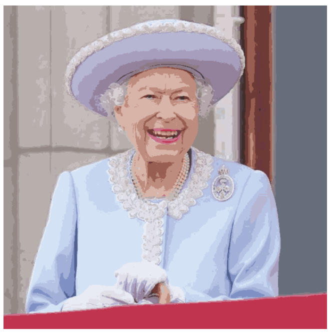
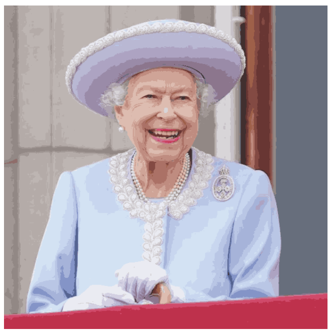

(d) sRGB Clustered image

Figure 2: Color palette extraction using the RGB and sRGB color spaces.


(a) Original image


(b) HSV color space


(c) CIELab color space

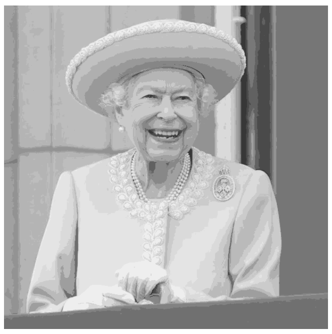
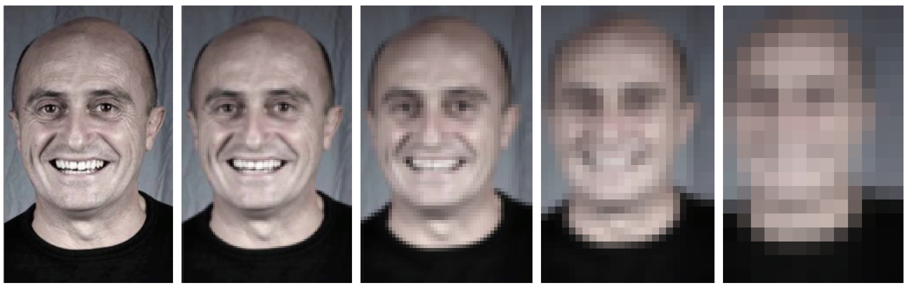

(d) Grayscale-mapped

Figure 3: Quantized images with 32 clusters (colors) in different color spaces

```matlab
function [gaussian_pyramid, laplacian_pyramid] = compute_pyramids(img, layers)
    layers = layers+1;
    gaussian_pyramid = cell(1,layers);
    gaussian_pyramid{1} = img;

    % compute gaussian pyramid
    for i = 2:layers
        prev_img = gaussian_pyramid{i-1};
        new_img = imgaussfilt(prev_img,1);
        new_img = imresize(new_img,0.5,'nearest');
        gaussian_pyramid{i} = new_img;
    end

    % compute laplatian pyramid
    laplacian_pyramid = cell(1,layers);
    for i = 1:layers-1
        prev_img = gaussian_pyramid{i};
        next_img = gaussian_pyramid{i+1};
        laplacian_pyramid{i} = prev_img - imresize(next_img,[size(prev_img,1),size(prev_img,2)],'nearest');
    end
    laplacian_pyramid{layers} = gaussian_pyramid{layers};

end


function [img] = reconstruct_from_laplacian(laplacian_pyramid)
    layers = size(laplacian_pyramid,2);
    img = laplacian_pyramid{layers};
    for i = layers-1:-1:1
        img = imresize(img,[size(laplacian_pyramid{i},1),size(laplacian_pyramid{i},2)],'nearest') ...
                + laplacian_pyramid{i};
    end
end
```

Figure 4: Matlab functions computing the Gaussian and Laplacian pyramids and reconstructing the original image.


(a) Gaussian Pyramid layers

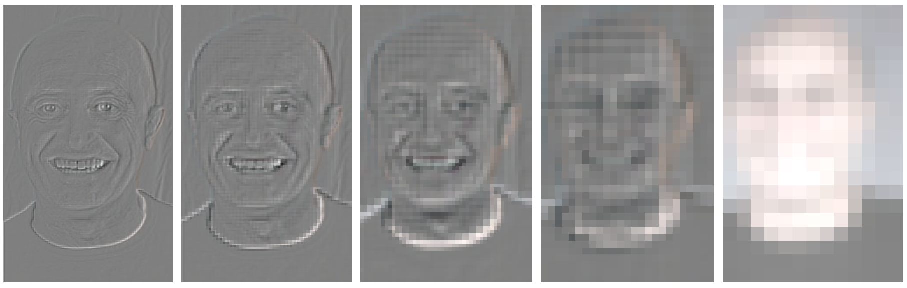

(b) Laplacian Pyramid layers (brightness artifically increased)

Figure 5: Image pyramid decomposition of `happy.jpg`

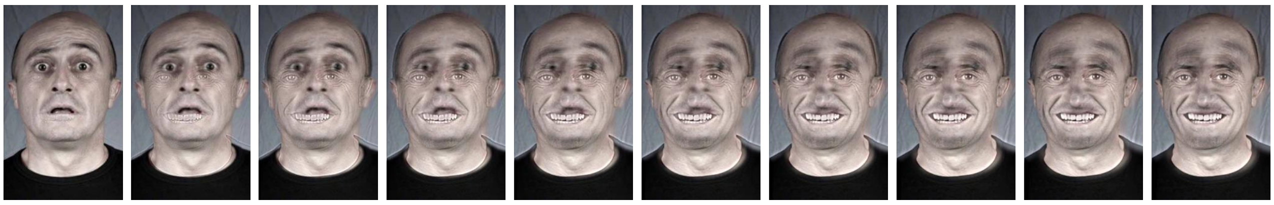


Figure 6: Hybrid images with different σ values. Lower-Higher σ from left to right.

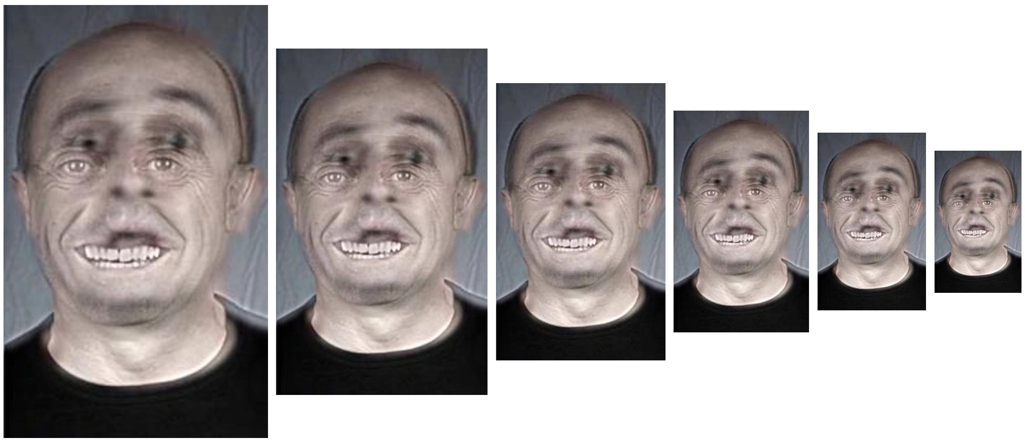
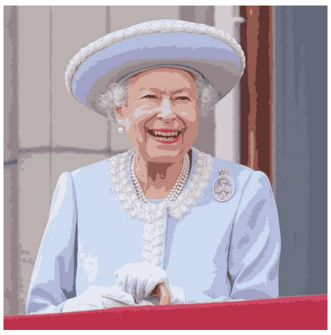


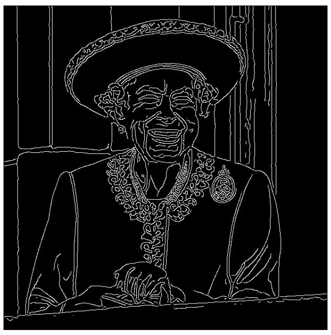
Figure 7: Showcase of frequency band sensitivity of the human visual system related to the image size.

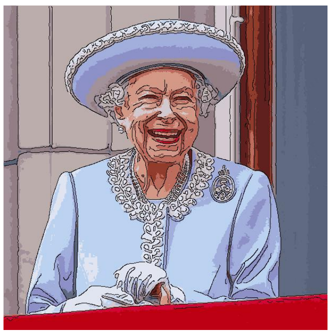
```
function [img, img_clustered, edges, im_stylized] = filter(img)
    % 1. Cluster the image colors
    clusters = 16;
    [img_clustered, cluster_idx, cluster_center ] = clusterColors(img, clusters);

    % 2. Find edges using Canny algorithm
    edges = edge(rgb2gray(img), 'canny', 0.1);

    % 4. Use the edges as a mask to stylize the image
    im_stylized = img_clustered.*~edges;

    % 5. Saturate the colors in HSV space
    im_stylized = rgb2hsv(im_stylized);
    im_stylized(:, :, 2) = im_stylized(:, :, 2)*1.5;
    im_stylized = hsv2rgb(im_stylized);

    edges = gray2rgb(edges, img);
    imshow([img, img_clustered, edges, im_stylized], []);

    function [img_clustered, cluster_idx, cluster_center ] = clusterColors(img, n_clusters)
        % Cluster
        A = reshape(img, [], 3);
        [cluster_idx, cluster_center] = kmeans(A, n_clusters);
        % Compute clustered image
        img_clustered = reshape(cluster_center(cluster_idx, :), size(img, 1), size(img, 2), 3);
    end
end
```

Figure 8: Matlab function applying the *Borderlands 2* filter.

(a) 1. Original image


(b) 2. Quantized image


(c) 3. Edges of quantized image


(d) 4. Masked and saturated final image